On the Use of Hard and Soft Reservation Schemes in Constant Bandwidth Servers

Alexandre Passos, George Lima

[1] Deparment of Computer Science (DCC)
Federal University of Bahia (UFBA)
Salvador, BA, Brazil

Abstract. *Reservation-based scheduling has successfully been used as a suitable technique to support complex real-time systems. In particular, the Constant Bandwidth Server (CBS) is one such technique that has received special attention recently. In this paper we evaluate by extensive simulation two CBS versions, corresponding to hard and soft reservation schemes. The latter makes use of background processor time but to do so it may decrease the server priority too much. While the former circumvents this problem, it follows stricter reservation rules. Results obtained from our simulation indicate that both schemes perform equivalently in some scenarios while in several others the soft version significantly outperforms the hard one in mean response time, response time jitter and deadline miss ratio. Such results are relevant for system designers who can better subsidize their implementation decisions.*

1. Introduction

Real-time systems once structured as a set of simple periodic hard tasks [Liu and Layland 1973] have become more complex. Nowadays such systems are often composed of hard and soft real-time tasks, both of which can be either periodic or non-periodic. Usual requirements of these modern real-time systems include not only predictability but also responsiveness and adaptiveness. In this context, reservation-based scheduling, **RBS** for short, has successfully been applied to support such requirements [Abeni and Buttazzo 2004, Mercer et al. 1994, Rajkumar et al. 1998, Sprunt et al. 1989, Steffens et al. 2003].

RBS techniques are usually implemented by scheduling servers. A server is a virtual task responsible for scheduling application tasks assigned to it. Each server S is determined by a processor reservation tuple, (Q, T), where Q is called the server capacity and T the server period. If a task (or group of tasks) τ is assigned to a server S, τ has the right to use Q processor time units within T total time units. Hence, servers can then be scheduled as if they were simple periodic tasks.

On top of its conceptual simplicity, RBS has several interesting properties [Steffens et al. 2003]. From the scheduling viewpoint it is worthing mentioning that: (a) RBS is a means of ensuring temporal isolation since task execution can be controlled within its reservation envelope [Abeni and Buttazzo 2004, Mercer et al. 1994, Sprunt et al. 1989, Spuri and Buttazzo 1996]; (b) it is possible to implement hierarchical scheduling when a group of tasks share the same server [Davis and Burns 2005, Davis and Burns 2008]; (c) it can be used to improve flexibility, responsiveness and adaptiveness at the scheduler level. Such characteristics have motivated considerable research in the area [Abeni and Buttazzo 1999, Caccamo et al. 2000, Caccamo et al. 2005,

de Oliveira et al. 2008, de Oliveira et al. 2009, Lin and Brandt 2005]. In this paper we aim at better characterizing the performance of two reservation schemes commonly implemented in real-time systems [PERNG et al. 2006].

A reservation scheme is **hard** if a server $S = (Q, T)$ cannot be scheduled to run for more than Q time units within any time interval of T units even if there is idle processor time available. On the other hand, a **soft** scheme allows S to execute beyond its reservation as long as system schedulability is not compromised. Often the decision of implementing one of these schemes arises. One of the widely known implementations of soft RBS scheme in EDF scheduled systems is the Constant Bandwidth Server (**CBS**) [Abeni and Buttazzo 2004]. Due to the number of other scheduling mechanisms based on CBS [Abeni et al. 2005, Caccamo et al. 2000, Caccamo et al. 2005, Lipari and Baruah 2000], we chose it as a case study in this paper.

According to the CBS rules, a constant bandwidth $u = Q/T$ is allocated to each server $S = (Q, T)$, where Q is called hereafter the maximum server budget. The budget of each server S is consumed as its jobs are executed. To guarantee that the server obeys its utilization limit, the deadline of S is postponed by T every time its budget is depleted. Its full budget Q is also restored when its deadline is updated. Deadline postponements make it possible to implement the soft reservation scheme but brings about a potential side effect: an overloaded server may have its deadline postponed too often making its relative priority too low. This problem may be avoided by implementing a hard reservation version of CBS [Buttazzo 2005], which waits until the current server deadline to recharge the server budget. In order to support implementation decisions regarding hard and soft reservation schemes comparative studies are still needed.

1.1. Related work

There is extensive research on comparing RBS techniques. Some of them are based on fixed-priority scheduling [Bernat and Burns 1999, Bernat and Burns 2002, Davis and Burns 2005, Davis and Wellings 1995]. In the field of dynamic scheduling, CBS has been compared to several other approaches [Spuri and Buttazzo 1996]. Techniques to reclaiming spare capacity of CBS have also been reported, bringing in comparative studies on their performance [Caccamo et al. 2000, Lin and Brandt 2005]. To the best of our knowledge these comparative studies have considered only the soft version of CBS, though. The work by Rajkumar et al. [Rajkumar et al. 1998] illustrates the use of hard and soft versions of a fixed-priority RBS using a very simple task set.

1.2. Contributions of this paper

We report a comprehensive comparative study on the performance of CBS. The reported evaluation puts into perspective the perceived characteristics of its hard and soft reservation schemes. By benchmarking these two CBS versions in a simulation environment, they are systematically compared against each other. Since the use of CBS has extensively been reported and one usually chooses arbitrarily between these two versions (e.g. the work by Abeni et al. on adaptive reservations and QoS [Abeni and Buttazzo 1999, Abeni et al. 2005]), the results reported here can better subsidize their implementation decisions. The reported study was carried out making use of the real-time scheduling simulator **RTSimul**, implemented in Pyton. Although there

are other simulation tools [Ancilotti et al. 1996], implementing this simulator ourselves allowed us to easily tune it to our specific requirements.

1.3. Structure of this paper

A more precise definition of the CBS versions considered in this paper is given in Section 2. Some comments on the motivation for this study is also presented in this section. Section 3 describes the system model and the simulation environment. In order to give a wide picture of the simulated scheduling mechanisms, we conducted the evaluation taking into consideration different simulation scenarios. Results from the simulation are discussed in Section 4 while in Section 5 our final comments are given.

2. Soft and hard CBS

A CBS [Abeni and Buttazzo 1998] is defined by a tuple (Q, T) and by a set of rules. The server is allowed to run for Q time units over a period of T time units. Whenever a job j arrives at the server, it is enqueued in the server queue. The server also maintains internally a tuple (c, d), where c is its current budget and d is its current deadline. For each time unit a task spends running, the value of c is decreased by the same amount. The CBS is scheduled as a normal task with deadline d running on an EDF scheduler.The CBS algorithm can be found in Abeni et al.[Abeni and Buttazzo 1998].

It is interesting to observe that the soft CBS trades availability for priority. Indeed, in the soft version, a server is always ready since its budget is never kept null. Nonetheless, the sever may have its deadline postponed too much in periods where it is overloaded. If this occurs, the server may wait in background for some time. On the other hand, the hard CBS does not suffer from this effect but may be kept suspended for some time. The characterization of these versions with respect to their effects on applications will be reported in the following sections.

3. Simulation environment

To perform the experiments presented in this paper we implemented **RTSimul**, a simple python-based simulator for real-time scheduling algorithms. Its source code, the data files used in this paper and instructions to reproduce our results are available in `http://github.com/jamsjr/hard-versus-soft/tree/master`. **RTSimul** implements a basic EDF scheduler and, on top of it, runs both hard real-time periodic tasks and bandwidth reservation servers. These servers can be either hard [Buttazzo 2005] or soft CBS [Abeni and Buttazzo 1998], as previously described. Soft real-time tasks running on these servers either have their execution times sampled from a probability distribution or follow traces generated from a modified version of the `FFMPEG` decoding library that reports the start time and decoding cost for each frame in a high-resolution video. The same time unit standard was used for both the video trace and the synthetic task sets so that all presented measurements are platform-independent.

Most simulations performed in this paper share a common set-up. There is a periodic hard real-time task and one soft real-time server, implementing either hard or soft reservation schemes. The server and the task compete for processor according to EDF rules. The server is monitored during the simulation so that both reservation schemes can be compared against each other. This simple simulation set-up makes it easier to

analyze the behavior of the system under different load and configurations. More complex simulations, involving more servers and tasks, were also tested. In both complex and simple simulation setups, the behavior of the reservation schemes was shown to be similar. In order to illustrate this, we also present results obtained by a more complex simulation set-up. In the following we define the configurations and the server loads considered during the simulation.

3.1. Simulation configurations

In order to design our simulation to cover several different application semantics, we have considered four basic load simulation parameters:

Discarding/not discarding expired tasks. For some applications it is better to discard tasks when they miss their deadlines while for other ones tasks are kept executing after their expired deadlines.

High/low execution cost variance. Execution cost variance plays an important role when tuning the server parameters. Server capacities are usually adjusted to the mean execution cost value. Varying this parameter we verify the behavior of the reservation schemes for these two extreme scenarios.

Overloaded/not overloaded system. We consider a system overloaded when its tasks demand more than 100% of CPU.

Overloaded/not overloaded server. A server S is overloaded when its utilization $u = Q/T$ is not enough to serve the tasks it is serving.

Considering all combination of these four parameters leads to 16 distinct configurations. In several of them the behaviors of both reservation schemes were found to be equivalent or their differences were not statistically significant. For the ones in which they are not equivalent, we grouped the possible configurations into five scenarios. The metrics we used to evaluate the performance of the servers are response time, wait time and deadline miss ratio.

3.2. Characterizing the server load

Five distinct application loads (*Simple*, *Time*, *Variance*, *Movie*, and *Complex*) were considered by varying the execution cost distributions. For the sake of data analysis, it is interesting to observe how hard and soft reservation schemes behave with more controlled load distributions. This is the goal of loads *Simple*, *Time*, and *Variance*, which takes synthetic generated data. The data correspond to execution times of jobs to be served during $9,000$ time units of simulation. The inter-arrival time of these jobs were kept constant and equal to one time unit. The load *Movie* was extracted from a movie trace and was used to represent more realistic soft-real time data. Load *Complex* uses the same trace as load *Movie* but runs 5 independent servers competing for the processor. The characteristics of these five load scenarios are given below. In all loads, the server capacity was set to the mean cost of a task, while the server period was set to the mean task period.

Simple: Constant mean execution time and low variance. Job execution costs were generated during the simulation time, set to $9,000$ time units, according to a normal distribution with mean 0.4 and (low) variance equal to 0.01. In other words, the execution cost of each job to be served follows $\mathcal{N}(0.4, 0.01)$. The background task, in this scenario, consumes 60% of the processor time. The server's capacity is 0.4 and its period is 1 time unit.

Time: Variable mean execution time and low variance. In order to simulate controlled variation of execution times, two values for the mean execution time were considered, sampled from $\mathcal{N}(0.6, 0.01)$ and $\mathcal{N}(0.4, 0.01)$. During the time intervals $[0; 3,000)$ and $(6,000; 9,000]$ data drawn from the former were used while the latter distribution was used to generate data for interval $[3,000; 6,000]$. This simulates scenarios were the application's execution can be partitioned into different phases, each with its particular demands of processor time. The background task, in this scenario, consumes 60% of the processor time. The server's capacity is 0.46 and its period is 1 time unit.

Variance: Constant mean execution time and variable variance. Another simulated scenario considered two simulation intervals, $[0; 4,500)$ and $[4,500; 9,000)$. The execution costs for these intervals followed two distributions, $\mathcal{N}(0.5, 0.01)$ and $\mathcal{N}(0.5, 0.1)$. The background task consumes 50% of the processor time, the server's capacity is 0.5 and its period is 1 time unit.

Movie: A movie trace. The execution cost distribution are shown in Figure 1. The values in the graph correspond to the decoding times using the `FFMPEG` video decoding library for the Eve movie. In this trace jobs arrive with a mean period of 0.072 time units and mean costs of 0.0064 time units. This trace is considered for the purpose of testing the behavior of both reservation schemes with a real-life data set. The background task consumes 90% of the processor time. The server's capacity is 0.0065 and its period is 0.072 time units.

Complex: A complex scenario. Five competing servers, each with the same task set as scenario *Movie*. Each server has capacity 0.0065 and period 0.072, and the background task takes 50% of the processor time.

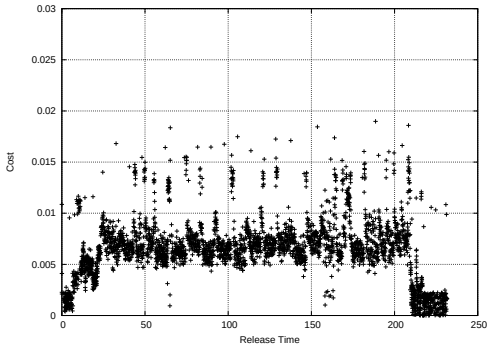


Figure 1: Execution cost distribution for scenario *Movie*.

4. Simulation results

As mentioned in Section 3.1, not all configurations led to a performance difference. Section 4.1 first presents some comments on the reasons for this behavior, and then, Section 4.2 describes the results that highlight any differences in performance between the hard and soft reservation schemes.

4.1. Equivalent reservation performance

In configurations where expired tasks are not discarded, the system is constantly overloaded or the server is underloaded both reservation schemes performed equivalently. The explanation for this behavior is given below.

4.1.1. Not discarding expired tasks

Consider a server $S = (Q, T)$ and that at time t there is an execution cost of C to be served. Since expired tasks are not discarded, both hard and soft reservation servers will finish in at least $\lfloor C/Q \rfloor T$ time units. The only possible difference is that a soft reservation server might spend this budget before a hard reservation server can. In that case, the soft reservation server may suffer some deadline postponements, but can never finish executing a given job after an equivalent hard reservation server. Hence, in general, not discarding expired tasks makes soft and hard reservation completely equivalent, and we will omit the results for the simulations that do not discard expired tasks.

4.1.2. Overloaded system

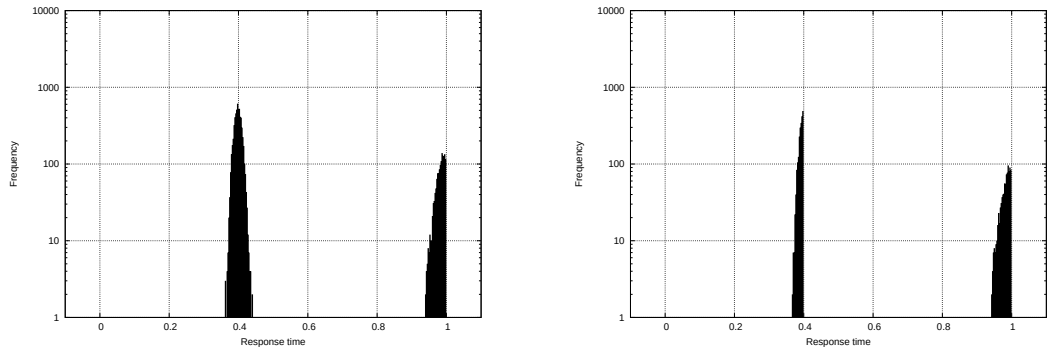
When the system is always perfectly dimensioned or overloaded (there is no system idle time, or, equivalently, all the demand for the servers plus the background tasks is more than 1) both reservation schemes are equivalent. This is so because a soft reservation server will only use extra budget obtained by postponing its deadline when there are no other tasks with a higher priority than its next deadline, and this situation is unlikely in an overloaded system. For this reason we only report the results of underloaded and correctly dimensioned systems.

4.1.3. Underloaded server

Another scenario that does not highlight any significant differences between hard and soft reservation is when the server load is low enough so that there is usually free budget upon completing a task. In these situations both hard and soft reservation mechanisms are equivalent, because, as less budget is needed per server period than the server capacity, there will be very infrequent budget exhaustion.

4.2. Unequivalent performance

Table 1 summarizes the simulation results for the previously described scenarios. In our simulations we do not monitor the background hard real-time task, since its hitting all its deadlines is guaranteed by our usage of the EDF algorithm and the system utilization being kept below 1. It can be easily seen that a soft reservation server has usually a much smaller deadline miss ratio than a hard reservation server in the same scenario. Also, the soft reservation server has a smaller jitter in all scenarios but *Time*, and even then the difference in standard deviation between the hard and soft servers is small. In scenario *Complex*, the table shows the mean response time, wait time and deadline miss ratio over all five servers in that scenario. The mean response times are mostly similar, except for scenarios *Movie* and *Complex*. It is important to note that comparing mean response time



(a) Soft reservation.

(b) Hard reservation.

Figure 2: Response time histograms for scenario *Simple*.

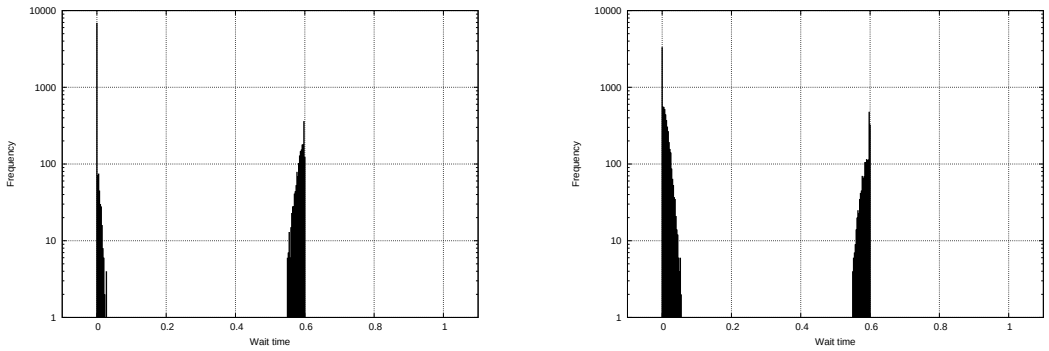
of servers with different deadline miss ratios is not reliable, since a server can always reduce its mean response time by failing to serve more demanding tasks. Hence, a more realiable picture of how the servers behave under each scenario can be pieced by looking at the mean wait time and the deadline miss ratio. These metrics show that a soft reservation server outperforms a hard reservation server in all simulated scenarios. The next sections go over each scenario in higher detail. All these scenarios are run discarding completed tasks.

LS	RS	MEAN RT	RT STDDEV	MEAN WT	DMR (%)
Simple	Soft	5.1×10^{-1}	2.3×10^{-1}	1.1×10^{-1}	5.4
	Hard	5.6×10^{-1}	2.6×10^{-1}	1.2×10^{-1}	59.8
Time	Soft	4.8×10^{-1}	1.2×10^{-1}	1.2×10^{-1}	14.4
	Hard	4.4×10^{-1}	1.0×10^{-1}	1.7×10^{-1}	37.5
Variance	Soft	5.7×10^{-1}	1.8×10^{-1}	8.0×10^{-2}	5.1
	Hard	5.8×10^{-1}	2.1×10^{-1}	1.2×10^{-1}	59.4
Movie	Soft	6.5×10^{-3}	4.8×10^{-3}	3.3×10^{-4}	0.1
	Hard	3.5×10^{-2}	3.1×10^{-2}	5.6×10^{-3}	24.5
Complex	Soft	2.5×10^{-3}	1.4×10^{-2}	6.4×10^{-4}	0.56
	Hard	3.8×10^{-2}	2.8×10^{-2}	1.2×10^{-2}	23.9

Table 1: Result summary, with the mean response time (RT), wait time (WT) and deadline miss ratio (DMR) given for each load scenario (LS) and reservation scheme (RS).

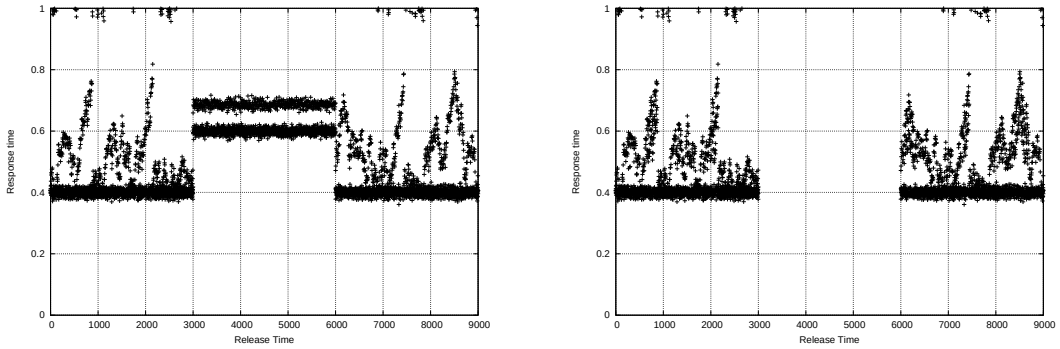
4.2.1. Scenario *Simple*

A general idea of how soft and hard reservation can behave might be given by a simulation with the costs in scenario *Simple*. Since the distribution of the execution costs is normal and constant throughout the simulation, the distribution of response and wait times is better characterized by a histogram. These are shown in Figures 2 and 3. In each graph, the horizontal axis represents each possible value, in time units, for the response or



(a) Soft reservation. (b) Hard reservation.

Figure 3: Wait time histograms for scenario *Simple*.



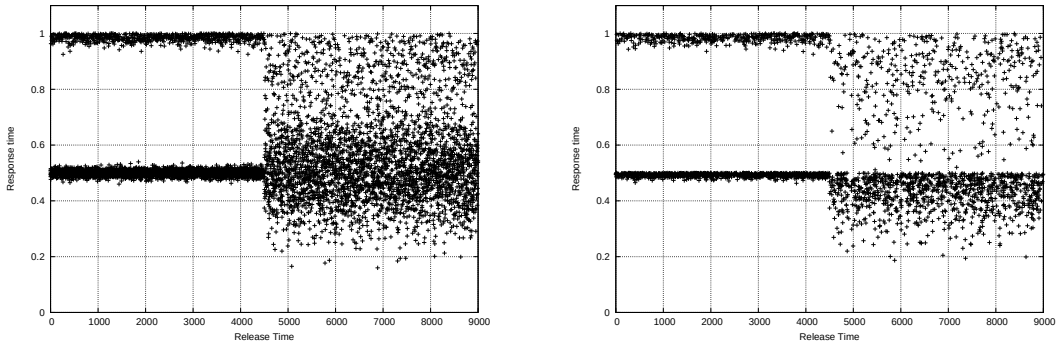
(a) Soft reservation. (b) Hard reservation.

Figure 4: Response times for scenario *Time*.

waiting times, while the vertical axis counts how many times each such value has actually occurred. In this scenario, the soft reservation server missed 5.4% of the deadlines and the hard reservation server missed 59.8%, as seen in Table 1. In Figure 2, the two modes of the distributions represent the response times for when the task executes immediately (the peaks around 0.4) and when it is delayed by the background task (the peaks around 1.0). Tasks that are delayed by more than that miss their deadlines.

4.2.2. Scenario *Time*

In this experiment, the task set first starts with a correctly dimensioned server, until time $t = 3000$, when the mean costs for the tasks rise suddenly. After a while, the costs return to normal. Figure 4 shows the response times for this scenario. The wait times, not shown specifically, follow a similar pattern. It is good to note that, as seen in figure 4b, all tasks in the higher demand section are discarded by the hard reservation server due to deadline missing. Each point (x, y) in the graph represents a task that, finishing at time x, had a



(a) Soft reservation.

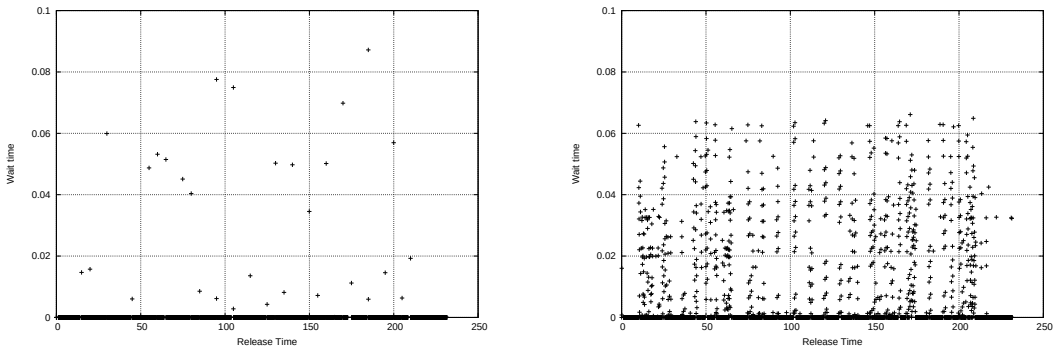
(b) Hard reservation.

Figure 5: Response times for scenario *Variance*.

response time of y. Therefore, when the same area is denser in one graph (as happens in the middle section of this scenario), it means that the server with a higher density completed more tasks. This kind of graph, while not precisely describing the distribution of the response times, shows a clear picture of how they change over time. As this figure shows, in the heavily loaded part only the soft reservation server can finish tasks that need more than the server budget. This suggests that the soft reservation server is better able to cope with run-time varying requirements. Also, when the load comes back down there is no extra cost for the soft reservation server, and its response times are equivalent to the response times for the hard reservation server. Thus, in variable environments a soft reservation scheme seems more suitable for situations where the application load changes over time. This information might be relevant for soft real-time systems running some adaptation mechanism, such as the one described by Abeni et al. [Abeni et al. 2005].

4.2.3. Scenario *Variance*

Another possible situation in which a RBS server might have its responsivity compromised is when dealing with tasks having a high variance in its execution costs. Scenario *Variance* was designed to test this possibility. This might happen because to serve tasks with unexpectedly high costs, a soft reservation server might need to postpone its deadline too much, while a hard reservation server would be more prudent and instead discard these costly outliers, being more performant in the remaining time. In this experiment the hard reservation server missed 59.39% of the deadlines, while the soft reservation only missed 5.19%. Figure 5 shows the response times for this experiment. The wait times behaved similarly and were omitted from this paper. As can be easily seen, in the higher variance part the soft reservation server manages to finish many more tasks in a short amount of time than the hard reservation server, thus behaving better on task sets with a high variance.



(a) Soft reservation.

(b) Hard reservation.

Figure 6: Wait times for scenario *Movie*, the movie trace.

4.2.4. Scenario *Movie*

Another very important test case is whether, in non-synthetic tasks, the performance characteristics we observed are reproduced. In this experiment, we ran both a soft reservation server and a hard reservation server on scenario *Movie*. Figure 6 shows the wait time for both a hard and a soft reservation servers running this task set. The response time follows a similar pattern and can be ommitted from this paper. As can be easily seen, the wait time for the hard reservation server is always higher than for the soft reservation server, sometimes high enough that the task has already missed its deadline by the time it starts executing. This might account for the fact that, as can be seen in Table 1, the soft reservation server only missed 0.1% of the deadlines, while the hard reservation server missed 24.54%. This result seems, at first, counter-intuitive, since the soft reservation server, by delaying its deadline whenever the execution costs are too high, can have an arbitrarily low priority. On the other hand, the same task will take more overall time to be served with a hard reservation server, since it will have to wait for capacity replenishment before continuing execution. Therefore, on average, the soft reservation scheme seems more responsive.

4.2.5. Scenario *Complex*

This scenario aims to show that increasing the complexity of the situation does not change perceptively the characteristics of hard and soft reservation. As can be seen in Table 1, the mean deadline miss ratio of the hard reservation servers is still substantially higher than the mean deadline miss ratio of the soft reservation servers. Indeed, the trends observed for scenario *Movie* are replicated.

5. Conclusion

Reservation-based scheduling has been increasingly used to support modern soft and hard real-time systems. Some implementations of both hard and soft reservation schemes have been proposed in real-time operating systems.

In this paper we have conducted a systematic study on the performance of soft and hard reservation schemes. As we have seen, both reservation schemes present equivalent performance for several configurations, but, using simple but illustrative load scenarios, we have shown that soft reservation schemes outperforms the hard one by having shorter response times, substantially shorter wait times and significantly smaller deadline miss ratios. These results suggest that soft reservation servers are better able to use system resources without either compromising overall system schedulability or task quality of service. We believe that the results and scenarios presented here can be used by designers to support their implementation decisions. This study can be extended by implementing soft and hard CBS servers in an actual real-time operating system, and using real applications as test data.

Acknowledgments

The dataset used in loads *Movie* and *Complex* was provided by Augusto Born de Oliveira. The second author is supported by FAPESB grant number 7320/2007. Some discussions with José Augusto Matos Santos in the beggining of this project were helpful. We'd also like to thank the anonymous reviewers for their helpful comments.

References

Abeni, L. and Buttazzo, G. (1998). Integrating Multimedia Applications in Hard Real-time Systems. In *Proceedings of the IEEE Real-Time Systems Symposium*, page 4. IEEE Computer Society Washington, DC, USA.

Abeni, L. and Buttazzo, G. (1999). Adaptive Bandwidth Reservation for Multimedia Computing. In *Proceedings of the IEEE Real Time Computing Systems and Applications*.

Abeni, L. and Buttazzo, G. (2004). Resource Reservation in Dynamic Real-Time Systems. *Real-Time Systems*, 27(2):123–167.

Abeni, L., Cucinotta, T., Lipari, G., Marzario, L., and Palopoli, L. (2005). QoS Management Through Adaptive Reservations. *Real-Time Systems*, 29(2):131–155.

Ancilotti, P., Buttazzo, G., Di Natale, M., Bizzarri, M., Anna, S., and Pisa, I. (1996). A flexible tool kit for the development of real-time applications. In *Real-Time Technology and Applications Symposium, 1996. Proceedings., 1996 IEEE*, pages 260–262.

Bernat, G. and Burns, A. (1999). New Results on Fixed Priority Aperiodic Servers. In *20th IEEE Real-Time Systems Symposium*, Phoenix. USA.

Bernat, G. and Burns, A. (2002). Multiple Servers and Capacity Sharing for Implementing Flexible Scheduling. *Real-Time Systems Journal*, 22:49–75.

Buttazzo, G. (2005). *Soft Real-Time Systems: Predictability Vs. Efficiency*. Springer.

Caccamo, M., Buttazzo, G., and Sha, L. (2000). Capacity Sharing for Overrun Control. In *Real-Time Systems Symposium, 2000. Proceedings. The 21st IEEE*, pages 295–304.

Caccamo, M., Buttazzo, G., and Thomas, D. (2005). Efficient Reclaiming in Reservation-Based Real-Time Systems with Variable Execution Times. *IEEE Transactions on Computers*, pages 198–213.

Davis, R. and Burns, A. (2005). Hierarchical Fixed Priority Scheduling. In *Proceedings of the 26th IEEE Real-Time Systems Symposium*, pages 389–398. IEEE Computer Society.

Davis, R. and Burns, A. (2008). An investigation into server parameter selection for hierarchical fixed priority pre-emptive systems. In *Proccedings of Real-Time and Network Systems, RTNS*.

Davis, R. and Wellings, A. (1995). Dual priority scheduling. In *Proc. of the 16th IEEE Real-Time Systems Symposium (RTSS '95)*, pages 100–109, Washington, DC, USA. IEEE Computer Society.

de Oliveira, A. B., Camponogara, E., and Lima, G. (2008). "Dynamic Reconfiguration of Constant Bandwidth Servers". In *Proceedings of the 10th Brazilian Real-Time Workshop (WTR 08)*, pages 1–8. Brazilian Computer Society.

de Oliveira, A. B., Camponogara, E., and Lima, G. (2009). "Dynamic Reconfiguration in Reservation-based Scheduling: An Optimization Approach (to apear)". In *15th IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS 09)*, pages 1–10. IEEE Computer Society.

Lin, C. and Brandt, S. (2005). Improving Soft Real-time Performance Through Better Slack Reclaiming. In *Proceedings of the 26th IEEE Real-Time Systems Symposium (RTSS 2005)*, pages 3–14.

Lipari, G. and Baruah, S. (2000). "Greedy Reclamation of Unused Bandwidth in Constant-Bandwidth Servers". In *Proc. of the IEEE 12th Euromicro Conference on Real-Time Systems (ECRTS 00)*, pages 193–200.

Liu, C. and Layland, J. (1973). Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment. *Journal of the ACM (JACM)*, 20(1):46–61.

Mercer, C., Savage, S., and Tokuda, H. (1994). Processor Capacity Reserves: Operating System Support for Multimedia Applications. In *Multimedia Computing and Systems, 1994., Proceedings of the International Conference on*, pages 90–99.

PERNG, N., LIU, C., and KUO, T. (2006). Real-time linux with budget-based resource reservation. *Journal of information science and engineering*, 22(1):31–47.

Rajkumar, R., Juvva, K., Molano, A., and Oikawa, S. (1998). Resource kernels: A resource-centric approach to real-time and multimedia systems. In *Proceedings of the SPIE/ACM Conference on Multimedia Computing and Networking*, pages 150–164.

Sprunt, B., Sha, L., and Lehoczky, J. (1989). Aperiodic Task Scheduling on Hard Real-Time Systems. *Real-Time Systems*, 1(1):27–60.

Spuri, M. and Buttazzo, G. (1996). Scheduling Aperiodic Tasks in Dynamic Priority Systems. *Real-Time Systems*, 10(2):179–210.

Steffens, L., Fohler, G., Lipari, G., and Buttazzo, G. (2003). Resource Reservation in Real-Time Operating Systems - A Joint Industrial and Academic Position. In *Proceedings of the International Workshop on Advanced Real-Time Operating System Services (ARTOSS)*, pages 25–30.